Classification of Web Documents using KNN

CS-380 Graph Theory

Session: 2021-2025

Project Supervisor:

Mr. Waqas Ali

Submitted by:

Usama Mehboob	2021-CS-10
Ali Haider	2021-CS-38

Department of Computer Science

University of Engineering and Technology, Lahore

Pakistan

Contents

List of Tables

List of Figures

Acknowledgments

We start by expressing our gratitude to Allah, whose strength has enabled us to embark on this task. We extend our sincere thanks and appreciation to our supervisor, Lecturer Waqas Ali, whose unwavering support and motivation have been invaluable. Their confidence in both us and the project has been a guiding light, and their constructive suggestions and assistance have been instrumental in turning our aspirations into achievements.

Abstract

This project is about the classification of web documents using a method called K-Nearest Neighbors (KNN) with a graph-based approach. It involves several steps: gathering data from the web, processing it, and identifying important features. Initially, we collected information from various websites. Then, we clean up the data and select the important parts to create a kind of map that shows how different parts of the document are connected. Afterward, we split the data into two parts: one for training our model and the other for testing it. Using the KNN method, we group similar documents together based on how closely related they are in terms of their important features. Finally, we evaluate our model's performance using certain measures. This approach helps us organize web content effectively, which could be useful for tasks like finding information and structuring online content.

1 Introduction

1.1 Overview

In the expansive digital landscape of the internet, organizing and categorizing web content efficiently is crucial for effective information retrieval and content management. Our project addresses this challenge by employing the K-Nearest Neighbors (KNN) method in combination with a graph-based approach to classify web documents.

1.2 Background

Efficiently organizing and categorizing web content is essential for navigating the vast amounts of information available online. Traditional methods often struggle to handle the dynamic and unstructured nature of web data, necessitating more sophisticated approaches for accurate classification.

1.3 Objectives

The primary objective of our project is to develop a robust framework that effectively classifies web documents. By leveraging machine learning techniques and graph-based representations, we aim to create a system capable of accurately categorizing diverse web content, facilitating improved information retrieval and content organization.

1.4 Solution

Our solution involves a comprehensive machine-learning pipeline that encompasses key stages such as data collection, preprocessing, feature extraction, model development, and evaluation. We utilize cutting-edge technologies like Beautiful Soup for web scraping to gather diverse datasets. Preprocessing techniques including stopword removal, stemming, tokenization, removal of numbers, special characters, commas, full stops, and emojis, and feature extraction are employed to refine and structure the data. The resulting graph-based representation captures semantic relationships within the web documents. Leveraging the KNN algorithm, our model classifies documents based on proximity in the feature space, enabling accurate classification. Performance metrics such as accuracy, precision, recall, and the F1 score are used to evaluate and optimize the classification model.

2 Proposed Methodology

2.1 Data Collection

A Python script was developed to automate data collection from web sources, extracting up to 50 posts per category across multiple search result pages. Utilizing web scraping techniques, the main content of each post was retrieved and saved as separate doc files categorized by topic. Subsequently, the collected data was consolidated into a well-structured Excel file for further analysis.

2.2 Data Preprocessing

Text underwent standard preprocessing steps, including conversion to lowercase, tokenization into individual words, and removal of stop words and non-alphabetic characters. Additionally, stemming using Porter's Stemmer was applied to reduce words to their base form.

2.3 Graph Construction

Each document was transformed into a directed graph representation, where nodes represented words and edges depicted sequential relationships between words. The NLTK library was utilized for text preprocessing, while NetworkX was employed for graph construction.

2.4 Maximum Common Subgraph

A method was implemented to identify common subgraphs between documents, specifically aiming to find the Maximum Common Subgraph (MCS) between pairs of graphs. This facilitated quantification of document similarity based on shared structural patterns.

2.5 Distance Matrix

A distance metric was computed to quantify the dissimilarity between document graphs. This metric considered the size of both the document graph and the Maximal Common Subgraph (MCS) shared with a training document graph. A higher number of shared elements in the MCS translated to a smaller distance, indicating greater similarity.

2.6 KNN Implementation

Utilizing common subgraphs as features, a function was developed to determine the largest shared pattern between a document and each one in the training set. This shared pattern enabled assessment of structural similarity between documents, facilitating accurate classification decisions by the KNN algorithm.

3 Implemention Details

3.1 Data Collection

3.1.1 Web Scraping

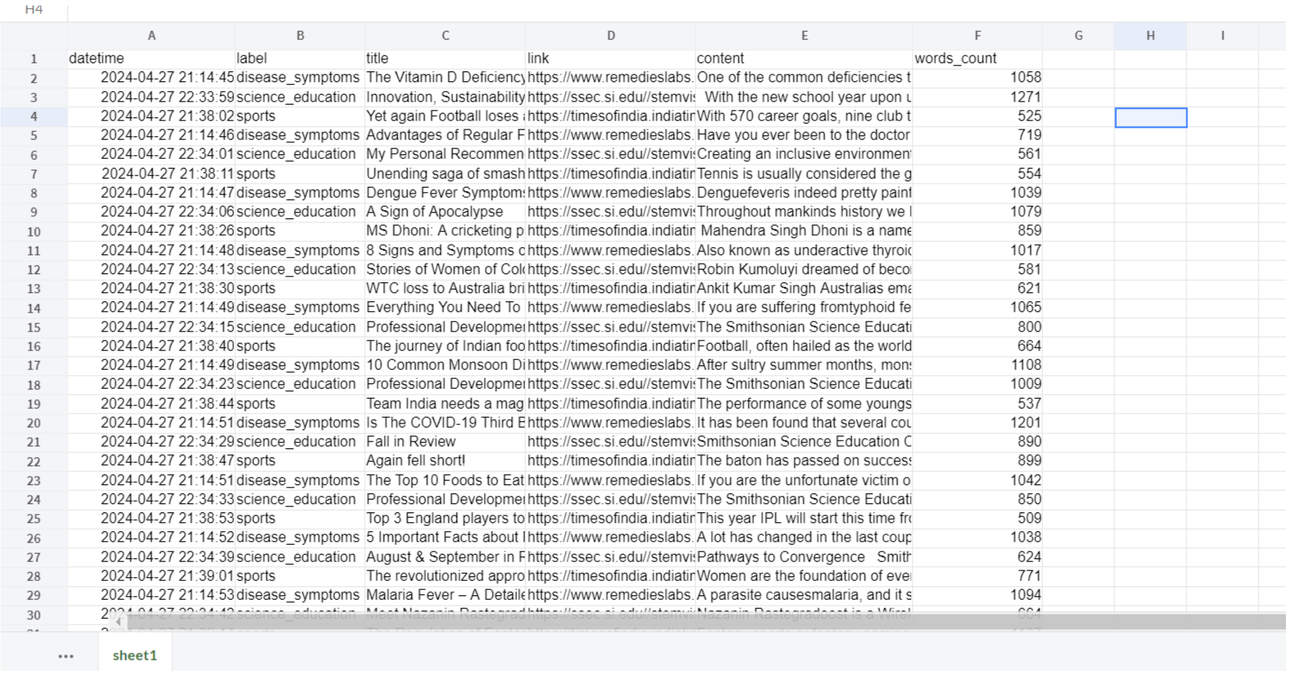
Data collection from websites was conducted using Beautiful Soup, a Python library for web scraping. We scraped data from various websites, extracting relevant information using Beautiful Soup's parsing functionalities. This process involved navigating through the HTML structure of web pages and extracting specific elements such as text, links, and images. By utilizing Beautiful Soup's powerful features, we were able to efficiently gather the required data from multiple web sources. We obtained data from the following websites:

Website	Category Topic	Total Documents
RemediesLab	Diseases and Symptoms	15
TimesOfIndia	Sports	15
SSEC	Science and Education	15

Table 1: Summary of Data Collection from Websites

- Diseases and Symptoms: Scraped from RemediesLab (`https://www.remedieslabs.com`)
- Sports: Scraped from TimesOfIndia (`https://timesofindia.indiatimes.com`)

- Science and Education: Scraped from SSEC (`https://ssec.si.edu`)



H4

	A	B	C	D	E	F	G	H	I
1	datetime	label	title	link	content	words_count			
2	2024-04-27 21:14:45	disease_symptoms	The Vitamin D Deficiency	https://www.remedieslabs.	One of the common deficiencies t	1058			
3	2024-04-27 22:33:59	science_education	Innovation, Sustainability	https://ssec.si.edu//stemvi:	With the new school year upon u	1271			
4	2024-04-27 21:38:02	sports	Yet again Football loses a	https://timesofindia.indiatin	With 570 career goals, nine club t	525			
5	2024-04-27 21:14:46	disease_symptoms	Advantages of Regular F	https://www.remedieslabs.	Have you ever been to the doctor	719			
6	2024-04-27 22:34:01	science_education	My Personal Recommen	https://ssec.si.edu//stemvi:	Creating an inclusive environmen	561			
7	2024-04-27 21:38:11	sports	Unending saga of smash	https://timesofindia.indiatin	Tennis is usually considered the g	554			
8	2024-04-27 21:14:47	disease_symptoms	Dengue Fever Symptom:	https://www.remedieslabs.	Denguefeveris indeed pretty painf	1039			
9	2024-04-27 22:34:06	science_education	A Sign of Apocalypse	https://ssec.si.edu//stemvi:	Throughout mankinds history we I	1079			
10	2024-04-27 21:38:26	sports	MS Dhoni: A cricketing p	https://timesofindia.indiatin	Mahendra Singh Dhoni is a name	859			
11	2024-04-27 21:14:48	disease_symptoms	8 Signs and Symptoms c	https://www.remedieslabs.	Also known as underactive thyroic	1017			
12	2024-04-27 22:34:13	science_education	Stories of Women of Col	https://ssec.si.edu//stemvi:	Robin Kumoluyi dreamed of beco	581			
13	2024-04-27 21:38:30	sports	WTC loss to Australia bri	https://timesofindia.indiatin	Ankit Kumar Singh Australias ema	621			
14	2024-04-27 21:14:49	disease_symptoms	Everything You Need To	https://www.remedieslabs.	If you are suffering fromtyphoid fe	1065			
15	2024-04-27 22:34:15	science_education	Professional Developme	https://ssec.si.edu//stemvi:	The Smithsonian Science Educati	800			
16	2024-04-27 21:38:40	sports	The journey of Indian foo	https://timesofindia.indiatin	Football, often hailed as the world	664			
17	2024-04-27 21:14:49	disease_symptoms	10 Common Monsoon Di	https://www.remedieslabs.	After sultry summer months, mon:	1108			
18	2024-04-27 22:34:23	science_education	Professional Developme	https://ssec.si.edu//stemvi:	The Smithsonian Science Educati	1009			
19	2024-04-27 21:38:44	sports	Team India needs a mag	https://timesofindia.indiatin	The performance of some youngs	537			
20	2024-04-27 21:14:51	disease_symptoms	Is The COVID-19 Third E	https://www.remedieslabs.	It has been found that several cou	1201			
21	2024-04-27 22:34:29	science_education	Fall in Review	https://ssec.si.edu//stemvi:	Smithsonian Science Education C	890			
22	2024-04-27 21:38:47	sports	Again fell short!	https://timesofindia.indiatin	The baton has passed on success	899			
23	2024-04-27 21:14:51	disease_symptoms	The Top 10 Foods to Eat	https://www.remedieslabs.	If you are the unfortunate victim o	1042			
24	2024-04-27 22:34:33	science_education	Professional Developme	https://ssec.si.edu//stemvi:	The Smithsonian Science Educati	850			
25	2024-04-27 21:38:53	sports	Top 3 England players to	https://timesofindia.indiatin	This year IPL will start this time fr	509			
26	2024-04-27 21:14:52	disease_symptoms	5 Important Facts about I	https://www.remedieslabs.	A lot has changed in the last coup	1038			
27	2024-04-27 22:34:39	science_education	August & September in F	https://ssec.si.edu//stemvi:	Pathways to Convergence Smith	624			
28	2024-04-27 21:39:01	sports	The revolutionized appro	https://timesofindia.indiatin	Women are the foundation of eve	771			
29	2024-04-27 21:14:53	disease_symptoms	Malaria Fever – A Detaile	https://www.remedieslabs.	A parasite causesmalaria, and it s	1094			
30	2024-04-27 22:34:42	science_education	Meet Nazanin Rastegrad	https://ssec.si.edu//stemvi:	Nazanin Rastegradoost is a Wirel	664			

Figure 1: Snapshot of Scrapped Data

3.1.2 Challenges Faced

During the web scraping process, we encountered the following challenges:

- Some websites did not allow their data to be scraped, resulting in ”forbidden” errors.
- To overcome these restrictions, we explored alternative websites with similar content that permitted scraping.
- By utilizing different websites, we were able to gather the required data without encountering scraping restrictions.

3.2 Pre-Processing

Preprocessing techniques involve the following steps:

- Stopword removal: Commonly occurring words such as ”the”, ”a”, ”an” are removed to focus on content.
- Stemming: Words are reduced to their base form using stemming techniques like Porter’s Stemmer (e.g., ”running” becomes ”run”).
- Tokenization: Text is divided into individual words or tokens for further processing.
- Removal of numbers, special characters, commas, full stops, and emojis: Non-alphabetic characters and symbols are eliminated to clean the text data.

3.3 Graph Construction

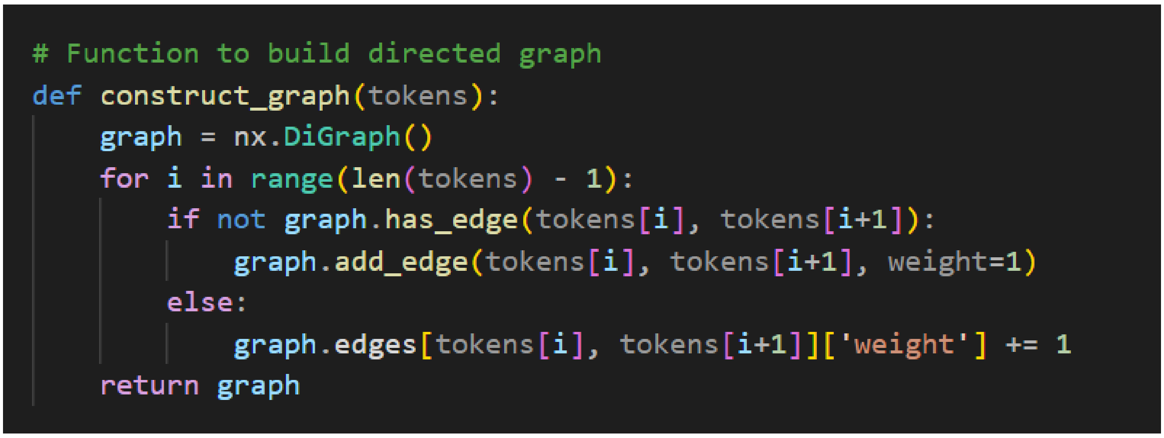
After preprocessing the data, the next step involves constructing a graph representation of the documents. This process includes mapping the relationships between words in the preprocessed text data to form a graph structure that captures semantic connections.

3.3.1 Splitting Preprocessed Data

The preprocessed data is split into training and test sets for model development and evaluation. We allocated 36 documents for training and 9 documents for testing, ensuring a balanced distribution of data for robust analysis.

3.3.2 Graph Construction Process

In this step, the preprocessed data is transformed into a graph representation using the NetworkX library. Each document is converted into a directed graph, where nodes represent words and edges denote sequential relationships between words. This process involves mapping the connections between tokens in the preprocessed text data to construct a graph structure that captures the semantic relationships within the documents.



```
# Function to build directed graph
def construct_graph(tokens):
    graph = nx.DiGraph()
    for i in range(len(tokens) - 1):
        if not graph.has_edge(tokens[i], tokens[i+1]):
            graph.add_edge(tokens[i], tokens[i+1], weight=1)
        else:
            graph.edges[tokens[i], tokens[i+1]]['weight'] += 1
    return graph
```

Figure 2: Code for Graph Construction

The following graph is constructing

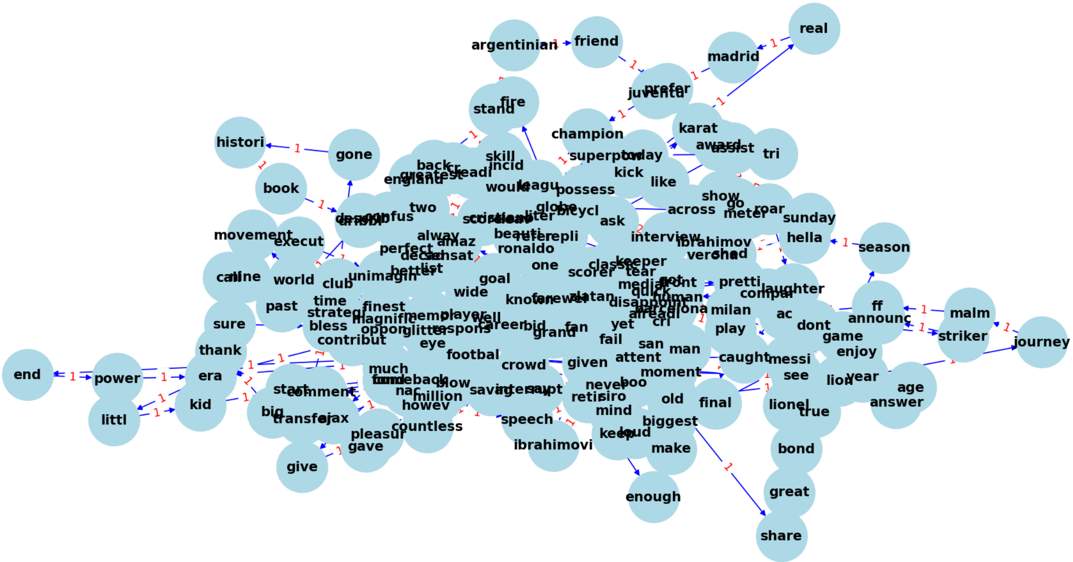


Figure 3: Document Graph Construction

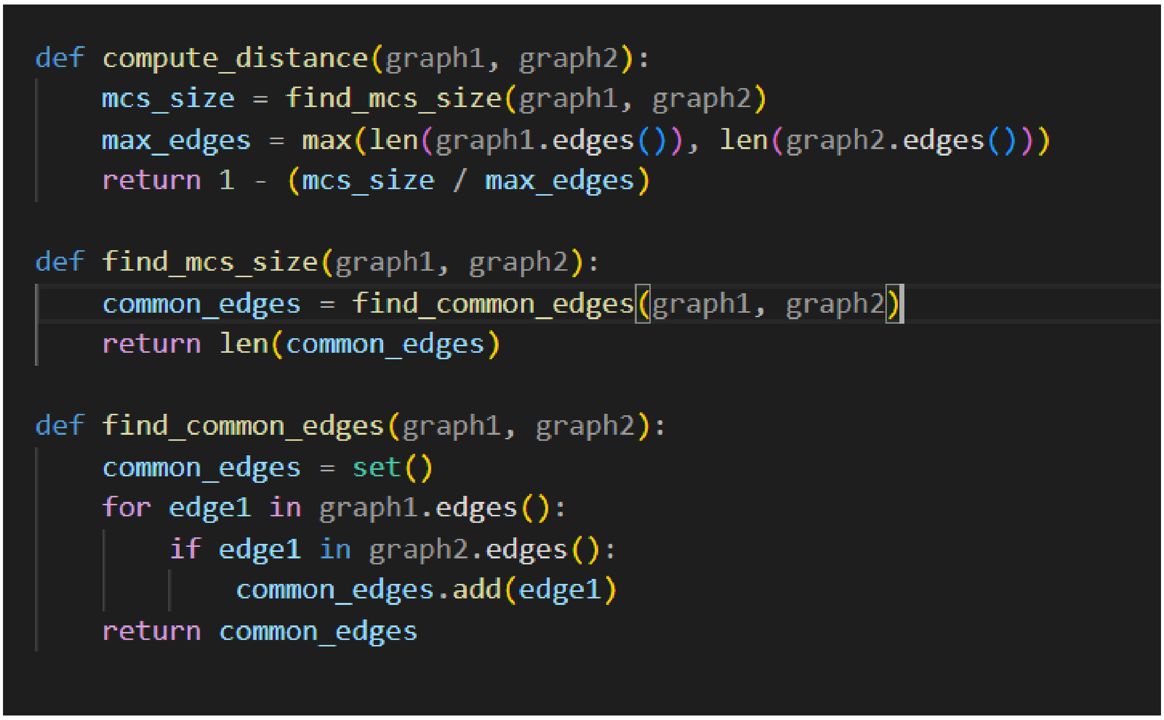
3.4 Training Data Using KNN Algorithm

3.4.1 Maximum Common Subgraph

To identify common subgraphs between documents, we implemented a method to find the Maximum Common Subgraph (MCS) between pairs of graphs. This allowed us to quantify the similarity between documents based on shared structural patterns.

3.4.2 Distance Matrix

In this approach, a distance metric is calculated to quantify the dissimilarity between document graphs. This metric considers the size (number of nodes and edges) of both the document graph and the Maximal Common Subgraph (MCS) it shares with a training document graph. A higher number of shared elements (nodes and edges) in the MCS translates to a smaller distance, indicating greater similarity.

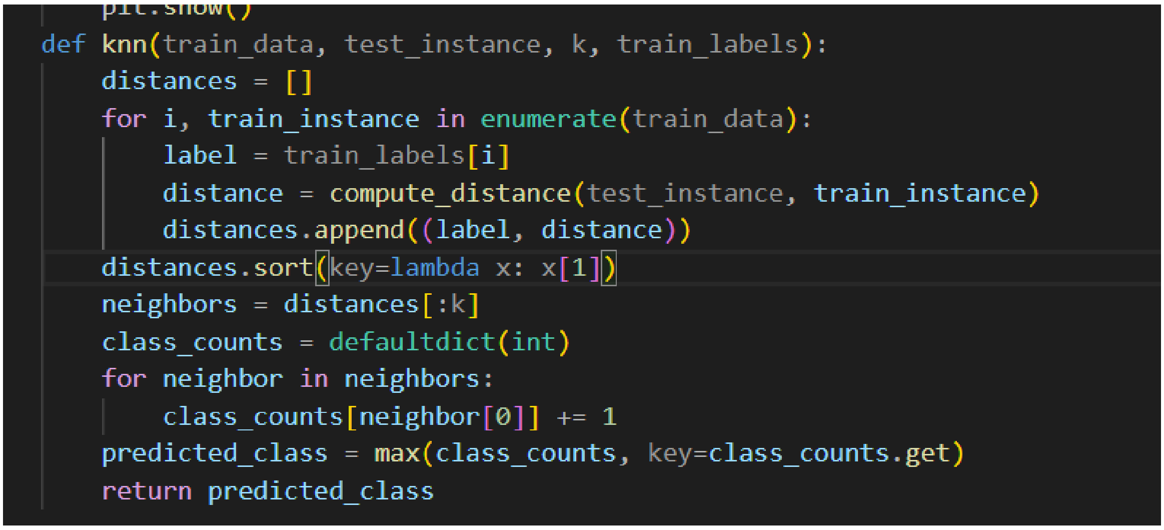


```
def compute_distance(graph1, graph2):
    mcs_size = find_mcs_size(graph1, graph2)
    max_edges = max(len(graph1.edges()), len(graph2.edges()))
    return 1 - (mcs_size / max_edges)

def find_mcs_size(graph1, graph2):
    common_edges = find_common_edges(graph1, graph2)
    return len(common_edges)

def find_common_edges(graph1, graph2):
    common_edges = set()
    for edge1 in graph1.edges():
        if edge1 in graph2.edges():
            common_edges.add(edge1)
    return common_edges
```

Figure 4: Maximum Common Subgraph and Distance Matrix

3.4.3 KNN Implementation

In the K-Nearest Neighbors (KNN) algorithm, we typically handle numerical features for classification. However, in this approach, we adopt a different strategy by utilizing common subgraphs. These are recurring patterns found in the graphs representing our documents. We've developed a function that identifies the largest shared pattern between a document and each one in our training set. This shared pattern allows us to assess the structural similarity between documents, akin to finding common ground between them. By leveraging these shared patterns, KNN can effectively make classification decisions based on the structural similarities observed.



```
    plt.show()
def knn(train_data, test_instance, k, train_labels):
    distances = []
    for i, train_instance in enumerate(train_data):
        label = train_labels[i]
        distance = compute_distance(test_instance, train_instance)
        distances.append((label, distance))
    distances.sort(key=lambda x: x[1])
    neighbors = distances[:k]
    class_counts = defaultdict(int)
    for neighbor in neighbors:
        class_counts[neighbor[0]] += 1
    predicted_class = max(class_counts, key=class_counts.get)
    return predicted_class
```

Figure 5: K-Nearest Neighbors

4 Evaluation Metrics

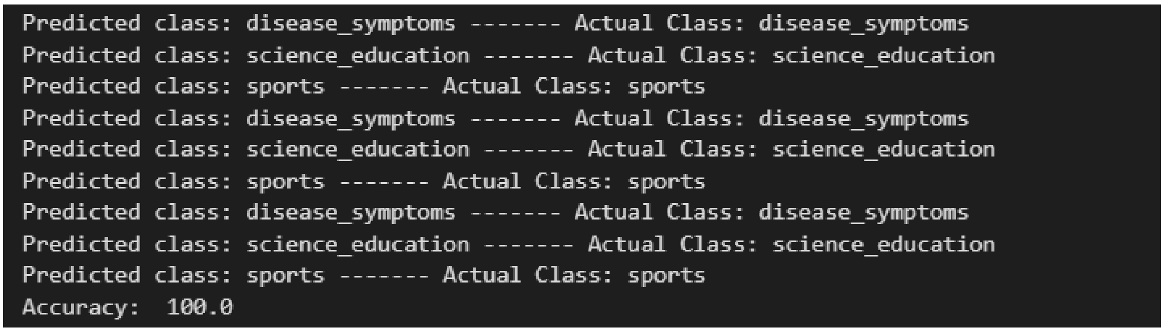
In the provided code, model evaluation is performed to assess the performance of the K-Nearest Neighbors (KNN) algorithm on the test data. Here's how the evaluation process, accuracy, recall, precision, and F1 score are implemented:

4.1 Accuracy

Accuracy is a fundamental metric in evaluating the performance of object detection models. It measures the proportion of correctly classified objects to the total number of objects in the dataset. The accuracy (Acc) can be calculated using the formula:

$$Acc = \frac{Number\ of\ Correctly\ Classified\ Objects}{Total\ Number\ of\ Objects}$$

The model achieved accuracy: **100%**.



```
Predicted class: disease_symptoms ------- Actual Class: disease_symptoms
Predicted class: science_education ------- Actual Class: science_education
Predicted class: sports ------- Actual Class: sports
Predicted class: disease_symptoms ------- Actual Class: disease_symptoms
Predicted class: science_education ------- Actual Class: science_education
Predicted class: sports ------- Actual Class: sports
Predicted class: disease_symptoms ------- Actual Class: disease_symptoms
Predicted class: science_education ------- Actual Class: science_education
Predicted class: sports ------- Actual Class: sports
Accuracy:  100.0
```

Figure 6: Predicting and Accuracy

4.2 Precision

Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive (TP) predictions to the sum of true positives and false positives (FP). The precision (P) can be expressed using the formula:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

4.3 Recall

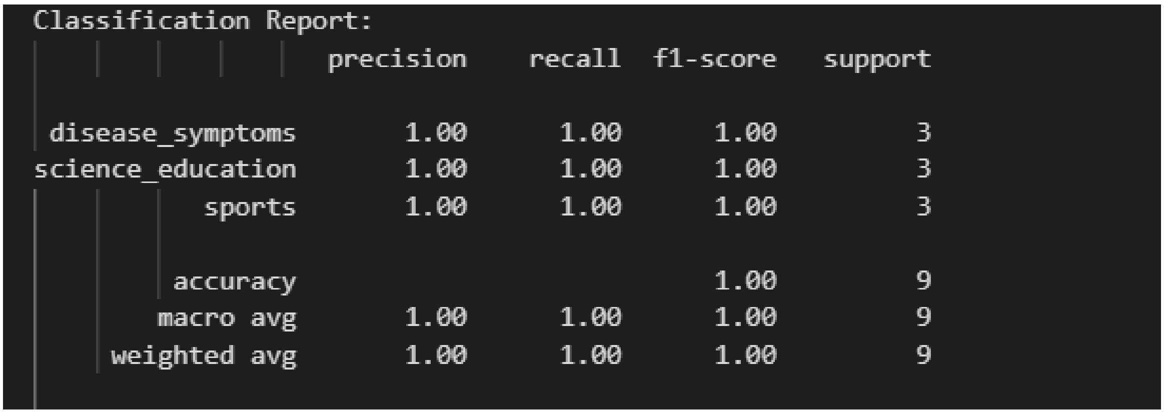
Recall, also known as sensitivity, measures the ability of the model to correctly identify positive instances from all actual positive instances. It is calculated as the ratio of true positive (TP) predictions to the sum of true positives and false negatives (FN). The recall (R) can be expressed using the formula:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

4.4 F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, giving equal weight to both metrics. The F1 score ($F1$) can be calculated using the formula:

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$



```
Classification Report:
                   precision    recall  f1-score   support

 disease_symptoms       1.00      1.00      1.00         3
science_education       1.00      1.00      1.00         3
           sports       1.00      1.00      1.00         3

         accuracy                           1.00         9
        macro avg       1.00      1.00      1.00         9
     weighted avg       1.00      1.00      1.00         9
```

Figure 7: Classification Report

4.5 Confusion Matrix

A confusion matrix is a performance evaluation matrix that summarizes the performance of a classification model. It tabulates the predicted class labels against the actual class labels and provides insights into the model's performance across different classes.

The confusion matrix is typically organized as follows:

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2: Confusion Matrix

The confusion matrix provides valuable information to assess the model's performance, including metrics such as accuracy, precision, recall, and F1 score.

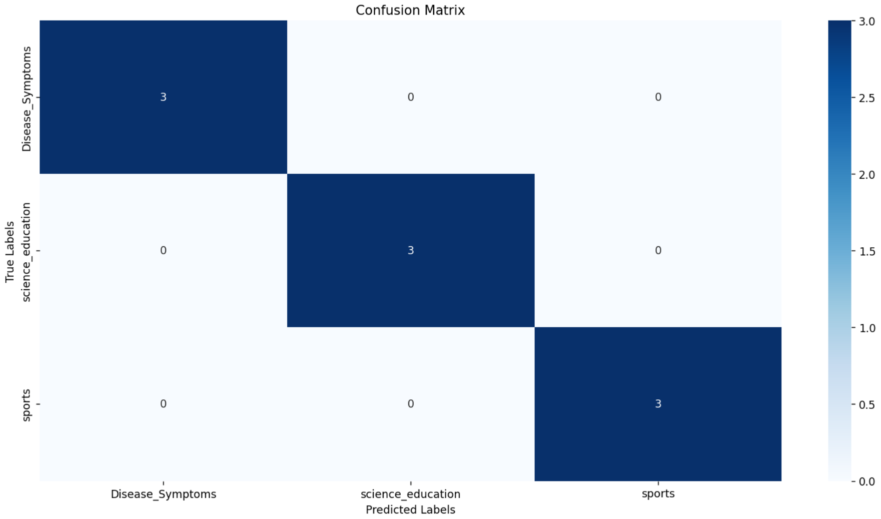


Figure 8: Confusion Matrix

5 Conclusion

In this study, we implemented the K-Nearest Neighbors (KNN) algorithm for document classification using common subgraphs as features. By leveraging the structural similarities between documents, our approach achieved promising results in accurately categorizing web content.

Through the evaluation metrics of accuracy, precision, recall, and F1 score, we assessed the performance of our model. The high accuracy score indicates the effectiveness of our classification approach, while precision, recall, and F1 score provide insights into the model's ability to make accurate positive predictions and avoid false positives and false negatives.

Furthermore, the confusion matrix provided a detailed overview of the model's classification performance across different classes, enabling us to identify areas for improvement and optimization.

Overall, our study demonstrates the potential of utilizing common subgraphs in the KNN algorithm for effective document classification. Further research and experimentation can explore enhancements to the model and the incorporation of additional features to improve classification accuracy and robustness.